COLOUR MONITOR 3.0

A Monitor Programme for the Colour Genie EG 2000 with 32K RAM

Written by Kaill Braun for TCS

COLOUR MONITOR 3.0

A Monitor Programme for the Colour Genie EG 2000 with 32K RAM

Written by Kaill Braun for TCS

(Note: this document was produced from a rough ORC scan I received via email. Some of the characters may not have reproduced faithfully so errors could be present Terry Stewart 25th July, 2009)

LOADING

Switch on and press <RETURN> in answer to MEM SIZE? Enter SYSTEM * ? Rewind the tape and press PLAY. Enter COLMON and press RETURN again. The programme should load.

If the two stars do not appear at the top right of the screen,or neither star blinks, or the left star changes to a C, interrupt the load by pressing both reset keys and reload at a different volume setting. If this doesn't work, try the second recording, which is on tape shortly after the first recording.

After a successful load, the computer responds with '*'. Type in '/' and press RETURN. The monitor is now in operation.

I Introduction.

Congratulations on purchasing the Colour Monitor 3.0. You have acquired not only the best CG monitor programme but one of the best monitors there are.

This monitor allows you not only to alter the memory of your computer byte by byte, but also to enter machine code programmes as mnemonics with the built in direct assembler. The monitor translates the mnemonics and stores them directly at the address you select. During the byte by byte editing you can switch from hex modification to ASCII input and have your modifications displayed directly disassembled. You can have your programmes executed in single steps and watch each change in the registers. This is only a brief look at the possibilities at your disposal. If you have problems or suggestions, write to me:

Halle Braun
Uhlgasse 11
5309 Meckenheim Merl
West Germany.

Best wishes for your success and enjoyment

II THE KEYBOARD

The use of the keyboard is quite normal Here are a few special points :

<CLEAR> During input to the assembler Start again
< > Cursor one space back

<RETURN> End input (at the beginning of a hex number)

<BREAK> Interrupt a command.

<*> Screenprinter (Screen dump to printer)

III COMMANDS

1) The Direct Assembler

Select <A> Address

The address into which the next command will be put is shown in the input line, followed by the cursor.

You have 5 possible inputs :

(a) < > decrease the address by 1
(b) < > increase the address by 1
(c) < > edit the last command
(d) < > jump a command
(e) Input a machine code command. You have these editing options:
< > deletes the last character
<CLEAR> deletes whole input
<RETURN> accepts the text inputted
Between the opcode and the operand there must be at least one space.
Permissible machine code commands are
all Z80 commands (see Zilog Z80 Assembly Language Program Manual, 3.0 D S.,REL 2.1, FEB 1977
DM Define Message The text beginning with the first non space character up to the end of the line is put into memory.
DW Define Word. The 16 bit Hex number following DW is put into the next two memory addresses. (LSB/MSB)
DB Define Byte. The byte following DB is put into memory.

If you input a non existent command, the assembler responds with 'Bad Opcode'. If you enter inappropriate arguments, (e.g. XOR 1) the assembler responds with 'Bad argument'.

After such error messages, press 'RETURN',

If everything is in order the inputted command is translated and stored at the address indicated. The assembler address is increased according to the length of the command and the computer asks for further input.

Example:

```
:8000 LD HL, (5800)
:8003 LD A,L
:8004  OR H
:8005 CALL NZ, 8010
:8008  JR 800
:800A <BREAK>
```

2. Setting Breakpoints

With this routine you may fix up to 11 addresses at which the single stepper (see also 10) can interrupt the execution of a programme.

Address are inputted and altered thus:

< > Change previous address

< > Change next address

<0> <F> Change the value indicated by the cursor

< > Cursor one place left

< > Cursor one place right

<RETURN> New address set the cursor at the beginning of the next line

<+> or <;> increase address by 1

< > decrease present address by 1

<X> delete present address

Example:

```
B       8000  <RETURN>
        1234  <RETURN>
        5800 <RETURN>
        <up arrow>
        <up arrow>
        <;>
        <up arrow>
```

If you have done everything according to the book, the file will look like this:

```
                1235

                5800
```

and the cursor will be on the '1'

3. Blanking memory areas

The command <C> really has two functions:

(a) Clearing breakpoints : <C> <RETURN> clears all breakpoint addresses set with (B).(see 2.)

(b) Filling memory areas'.

<C> Address 1 : fills memory from address 1 to FFFH with OOH <C> Address I address 2 bytes: fills the memory from address 1 to address 2 with 00H

<C> Address 1 address 2 byte: fills the memory from address 1 to address 2 with byte.

Example: C

```
C8000
C8000 GOFF 41
C8000 BOFF
```

4. HEX / ASCII DUMP.

With this command you can have memory contents listed in hex and ACSII. The command <D> can be used in two ways:

(a) <D><RETURN> Register Memory pointer on/off. In screen lines 14 17 you will see 8 bytes at a time the contents of the HL,DE,BC, and PC registers.

(b) <D> address In lines 18 25 of the screen 64 bytes are shown, starting from the address you input.

5 THE FIND DISASSEMBLER.

With this command you can search for a command in a programme. Give an address from which the search is to begin, and a machine code command you want to find. Then the search continues until either the command is found or the address 000H is reached

Example

```
E
E8000 LD HL,81FF
```

6. The BYTE SEARCH command.

This command searches for a byte sequence in memory. The difference between this command and <E> is as follows The <E> command looks for commands i.e if you are looking for the command LD HL,8000H, and at address 8OOOH there is LD IX, 9000H, the search continues at address 8004H, since the command LD IX, 9000H is 4 bytes long. The byte search command on the other hand moves on only one byte. So if you are looking for a 10 byte long sequence in memory, and the ten bytes from 8000H to 8009H do not correspond to the 10 bytes you are looking for, then the 10 bytes from 8001H to 800AH are compared, and so on.

Example:

```
Example: F8000 00 00 00
         F8000 01 02,03     04 50
```

7. Finding Programs

With this command you can start a programme in memory. If you input <G> only, you will jump to the address in the PC register. If you input <G> address, you will jump to the indicated address.

8 HEX ARITHMETIC

This command converts two hex numbers into decimal and outputs in two forms first in the range 0 (0000H) to 65535 (FFFFH) and then from 32768 (8000H) to 32767 7FFFH). The sum, the difference, the product and the quotient of the two numbers is also given in hex.

Example:

H7000 8000

7000 = 28672
= 28672

8000 = 32768
= 32768

7000+8000 = F000
7000 8000 = F000
7000*8000 = FOOO
7000/8000 = 8000

9 RELOCATING PROGRAMMES

With this command you can solve a great problem that confronts every programmer working with a monitor relocating programmes in memory.

If, for example, the command JP 9000H appears in your programme, the programme crashes if there is nothing at 9000H. If you have written a programme which occupies 9000H to A000H in memory, and you shift it to B000H, this can happen.

So you must look through your programme for such jumps and adapt them to the new addresses. This job is what the relocator does for you. You just have to input where the programme is to begin and end and by how much it is to be displaced. To shift a programme from 9000H to 8000H input FOOOH, (as 9000H+FOOOH is 8000H) and it is adjusted. Of course it is only adjusted, not shifted you must now do that with the Y command.

After inputting the offset you can input two further addresses for the following reasons. Normally, when you relocate, the only addresses which are changed are those in the memory area which is occupied by the programme. In most cases this is O.K. But what about when a programme is in an area of

memory in which it can't be run, because all the addresses still have to be changed? This would be too much for most relocators, and you would have to change all the addresses by hand.

But with this relocator it is possible to input two addresses as fourth and fifth argument, so that all the addresses in the area described by these two addresses are adjusted, and not in the area occupied by the programme itself.

So if you have a programme executable at A000H, but which is now at 7000H, and is to be executed from that address, you only need to input I7000 7FFF D000 A000 AFFF. Then D000H is added to all the addresses in the programme between A000H and AFFFH, so that the addresses are now between 7000H and 7FFH, as was necessary.

After inputting all necessary arguments, you are asked for one more letter code, (N,S,or C). With this letter you may preselect one of three options:

N Relocate wihout pause.

S Pause after each command you can go back one byte (up arrow) or forward one byte (down arrow). If you press <RETURN> the next command is relocated. This is especially useful when you have a programme in which there are a few texts at widel y spaced intervals. The relocator naturally interprets these texts as commands, and it may happen that texts are altered, as they may correspond to a command which needed to be altered. Besides this, it can happen that the last byte of a text corresponds to a command to which several arguments are attached. This would lead to commands being interpreted as arguments. To avoid this confusion, it is useful to be able to relocate such a programme with the option <S>.

C Pause at every command which has to be altered. By pressing (Y)es or (N)o you can decide whether the change is executed or not. As in option <S> this is useful when there are texts in the programme which are not to be relocated.

Example:

17000 7FFF D000 A000 AFFF N

10 STARTING THE SINGLESTEPPER

A programme is run from the address at which the PC is pointing and single stepped with regard for the options selected by the <T> command.

11 SINGLE STEPPING TO 'RET'

A programme is run from the address pointed to by the PC. The procedure is interrupted when the Command RET is reached

Example: K

12 Loading SYSTEM tapes

A machine code programme is read into memory from tape. If this programme is to be read into, a different area of memory than that indicated on the tape, you can input a value as an offset, which is added to the address on tape. On the completion of the load, the address at which the programme begins in memory, the address at which it ends, and the address to which you must jump to start the programme, are shown. The name of programme is also displayed.

Example:

L
TE000

13 MODIFYING MEMORY

With this command you can modify memory contents directly in hex and ASCII. By inputting the address at which YOU want to begin,

You can now modify memory in hex. In addition the following special key commands can be used:

Arrow keys: Move the cursor up, down, right and left. <F1> Note the present address

<F2> calculate the difference between the present address and the last address noted with <F1>, and file at the address noted with <F1>. (important for relative jumps)

With <CLEAR> you can switch to ASCII editing. The cursor now flashes to the right of the Hex/ASCII listing and you can enter Characters directly from the keyboard. Control of the cursor is above. By pressing <CLEAR> again, you can again edit in Hex.

Example:

M8000

14 EXECUTING COMMANDS

This routine enables you to enter machine code commands and execute them. For example:

You have written a subroutine and want to test it. Normally you would write a second programme, to call your subroutine, and then jump back into the monitcor. This roundabout method is now unnecessary. By using the <N> command you simply execute a call to your subroutine.

ExampIe:

```
NLD HL,8000 NCALL 7000
```

15 FINDING THE LOADING AREA OF A SYSTEM TAPE

This command is almost identical with the <L> command, but the data stored on tape is not loaded into memory, but a check is merely carried out as to where the data would be loaded with the normal <L> command. This is very helpful when you want to load a programme and don't know if it will overwrite the monitor.

The <O> command tells you where the programme would load without offset. Now you can calculate an offset which will load the programme into the memory area you want.

16 ENTERING A SINGLESTEP SUBROUTINE

Here you can enter the address of subroutines which are to be run with single step execution of a program.

Often the machine code programmer has the following problem:

A programme calls two subroutines, the first of which traverses a very long loop. This first subroutine works, but the second one causes a crash. What to do? It would take a long time to execute the first subroutine by singlestepping, and it would not be reasonable, as this programme is working well anyway.

You must only enter the starting address of the defective subroutine into the table, then use the <T> command to choose the appropriate single stepper option. Then you can single step your programme. The first routine is executed, but the second one is single stepped.

Entries into to the table correspond to the key functions described for the <B> command.

Example:

P

8000

8034

17 THE DISASSEMBLER

With the disassembler you can translate programmes in Your CG memory back into assembly language mnemonics.

This is essential if you want to analyze programmes or find mistakes. If you start the disassembler with 0 address, the next 11 commands after that address are displayed in the centre area of the screen. You now have the following option:

<RETURN> next 11 commands

< > previous 11 commands

<+> or <;> next command

< > 1ast command

<down arrow> + one byte

<up arrow> byte

<space> enter new address

The following is the second option of the <Q> command:

If you use the <Q> command without an address, then the 11 commands starting from the PC register in memory are listed. This is very useful with the single step execution with <S> and <X>, as you are then better able to follow the programme. If you use the disassembler like this, the environment of the address to be modified next is listed when you enter the assembler <A> command or hex/ASCII <M> command. This continuous disassembler can be switched off again thus:

by using the <B>, <H>, <L>, <O>, <P> or <W> commands

by pressing the <CLEAR> key

ExampIe

Q8000
Q

18 CHANGING THE CONTENTS OF REGISTERS

The register values displayed on the upper part of the screen are taken into the registers before every jump into a programme. Often it is useful to execute a subroutine separately, but this subroutine necessitates several arguments in different registers. The <R> command is useful here. It enables you to change register contents simply.

After selecting the <R> command, enter the name of the register you want to change and press the spacebar. (Admissible register names are: B,B',C,C’,D,D’E,E’,H,H’,L,L’,A,A’,F,F’,BC,BC’,DE,DE’,HL,HL’,AF,AF’SP,PC,IX,IY). Then enter the new value of the register and press <RETURN>.

Example:

```
RA  00

AF' FF RHL
7FFF
```

19 EXECUTING A COMMAND

By pressing the <S> the command to which the PC register is pointing is carried out, and the PC register is adjusted according to the length of the command. JPs, JRs etc cause a change in the PC register according to their argument, if the indicated conditions are met, otherwise the PC register is positioned behind the jump command. CALLS and RSTs are stepped through i.e. if the condition is met, a call leads to popping the jump back address from the stack and to a change of the PC register according to the argument of the CALL or RST command.

20 SINGLE STEPPER OPTIONS

With this command you have several options, width which you can influence the execution of a programme by the single stepper. In detail, these, options are:

Set breakpoints.

If YOU choose Me s here, the execution of a programme is interrupted when one of the addresses is reached which was entered with the <B> command. Enter <N>o, and these addresses are ignored.

Single Step RSTs.

Entering <Y>es causes every RST command in the programme to be executed in single steps. Entering <N>o causes an RST command to be executed as a command.

Single Step CALLS.

As above, but relates to all CALL commands in a programme (CALL and CALL conditions)

Single Step Rom CALLS

Enter (Y) and al I ROM CALLS .are executed in single steps, even if in the previous option you have answered <N>o.

Single Step Subroutines.

In this case (Y)es means that the subroutine indicated by the <P> command is to be executed in single steps, even if subroutines in general are not being single stepped.

Display Registers.

If you answer <Y>es here the state of the registers is displayed after every command.

Display Memory.

Answer (Y)es and after every command the memory area selected by the Hex/ASCII indicator on the lower part of the screen is updated.

- Number of instructions

Here you may enter how many commands are to be executed (entering 0000 means that all commands are to be executed)

Save Screen

Here you can enter an address from which the screen contents are to be saved after every command. This is important when your programme is supposed to produce a picture, and it doesn't work. If you now start your programme and return to the monitor after a breakpoint, the picture is destroyed. If at this point you indicate a buffer, the screen contents will be transferred there, before the monitor destroys the picture. An entry of 0000 means that the screen is not to be saved.

Execution Delay.

A value by which the execution of the programme is slowed down. 0000 means maximum delay, FFFF means minimum delay.

Example:
T

21 EXECUTING A 'RET'

The function of the RET command is simulated. The PC and SP registers are corrected.

Example:
U

22 COMPARING MEMORY AREAS

With this function you can compare the contents of two memory areas. You must enter the starting and finishing addresses the first area and the starting address of the second area. Then the comparison takes place. At any discrepancy the procedure halts, and above the Hex/ASCII clump a line from each memory area is displayed. The first bytes in each line are the two which do not correspond. If you now press <RETURN> the comparison continues.

Pressing the<BREAK> key ends the procedure .

23 WRITING A SYSTEM TAPE FROM A MEMORY AREA

With this command you can store your programmes on tape. You must enter the start address, the end address, and the execution address. Then you are asked for the programme name. This must not exceed 6 characters in length and must begin with a letter. When you have done this, press <RETURN> and recording will begin. Therefore, before pressing RETURN you must start your recorder in record mode.

Example:

W8000 87FF 8200

24 EXECUTING A COMMAND

This command is Similar to <S>, but all CALLS and RSTs are as not single stepped.

Example:

X

25 SHIFTING MEMORY AREAS

With this command your can shift a block of memory to another address. This is necessary , for example, when you want to shift a relocated programme to its execution address. Input the start and end address of the block to be shifted, and the address to which the block is to be shifted, then press <RETURN> to make the shift.

Note: The source area and the target area may overlap the shift will nevertheless take place correctly.

Example:

Y8000 8FFF 8800

Y8000 8FFF 6000

26 DISSASSEMBLE TO A PRINTER

If you have a printer connected to your CG you can print out a disassembled listing. Input the start and end address of the area YOU want to disassemble. The monitor will list the programme with a left margin of 10 spaces, with 60 line per page. At the head and foot of the page it leaves a marqin of 6 lines.

Eample:

Z8000 8156

Z0000 3FFF

27 HEX/ASCII DUMP 64 bytes back:

The address from which 64 bytes of memory are listed at the bottom of the screen is decreased by 64 bytes (one page back:) 0

Example:

28 HEX/ASCII DUMP 64 bytes forward

As above, but the displayed addresses are increased by 64 (one page forward)

Example
;
+

29 DECREASE C by 1

The PC register is decremented by 1

Example
(UP Arrow)

30 INCREMENT PC by 1

The PC is incremented by 1

Example:
(DOWN Arrow)

31 SET THE PC the next command

The command pointed to by the PC register is jumped.

Example:
(spacebar)

32 SET THE PC to the last command

The PC is setback one command, so that it points to the command before the one to which the PC register previously pointed

Example:
:

33 CLEAR SCREEN

Pressing the <CLEAR> key clears the middle part of the screen and switches off the permanent functions of the <D> and <Q> commands (see <D> and <Q> commands.

Example:
<CLEAR>

34 HEX/ASCII DUMP 1 byte back

The address from which 64 bytes of memory are listed at the bottom of the screen is decreased by one byte.
Example:
< left arrow >

35 HEX/ASCII DUMP 1 byte forward.

Likewise, but one byte forward.
Example:
< right arrow >

36 SHOW SECOND SCREEN

If when using the single stepper option you have set up a buffer, (see <T> command) where the screen contents are to be saved, you can bring back the contents of this buffer with this command. The screen remains unchanged until you press <RETURN>, when it is overwritten with the monitor working screen.
Example:
.

37 HEX/ASCII DUMP whole screen.

This command increases the section of memory which the HEX/ASCII dump produces on screen. Instead of 8 lines, 25 lines are displayed 200 bytes. In this mode you can go forward 200 bytes (<+> or <;> key) or back 200 bytes
  < > key. By pressing <BREAK> you get back into the normal mode.
Example:

/

38 DELETE THE SECOND SCREEN

If you set a buffer in the single stepper option, into which the screen contents are to be saved, (see <T> command), this area of memory is not blanked out. So you can fill it with blanks with this command.
Example:

39 SINGLE STEPPING 1 10 commands

With the 1 0 keys you can single step 1 to 10 commands one after another. After the execution of the inputted number of commands, the monitor stops again. All selected options are valid. (see <T> command).
Example:

0
5

IV APPLICATION EXAMPLE

Inputting a programme of your own. We now want to use the Monitor to input a programme:

```
8000 LD HL,4400
8003 LD DE,4401

8006 LD BC,03FF
8009 LD (HL),7F
8OOB LDIR

8OOD CALL 0049 8010
RET
```

This little programme paints the screen white, and then awaits a key input (CALL 0049). Then you jump back into the calling up programme. Shall we now enter this programme? There are basically two possibilities:

We translate the programme into hex code by hand and enter it with the <M> command:

```
M8000

21 00 44 11 01 44 01 FF 03 36 7F ED BO CD 49 00 C9

<BREAK>
```

or we enter the programme by using the built in assembler.

```
A8000

LD HL,  4400     <RETURN>
LD DE,  4401     <RETURN>
LD BC,  03FF     <RETURN>
LD  (HL),7F      <RETURN>
LDIR             <RETURN>
CALL  0049       <RETURN>
RET              <RETURN>
<BREAK>
```

However you entered the programme, you may now view it with the <Q> command:

```
Q8000

8000  21    00    44    LD HL,      4400H
8003  11    01    44    LD DE,      4401H
8006  01    FF    03    LD BC, 03FFH
8009  36    7F          LD (HL),7FH
800B  ED    BO          LDIR
800D  CD    49    00    CALL 0049H
8010  C9                RET
```

We leave the disassembler with <BREAK>
Now we set the PC to the beginning of our little programme: R PC 8000
and switch on the permanent function of the disassembler:

Q

We will now execute the little programme in single steps. Press the <X> key at every command:

X (The value of the HL register changes)
X (The value of the DE register changes)
X (The value of the BC register changes)
X (In the upper left corner of the screen a white block appears briefly)
X (The whole screen turns white briefly)
X (The computer waits in a loop until you press any key)

We see that our programme is working as we intended. Now we'll execute it in real time. So we set the PC register at an address where nothing can be destroyed:

R PC 4800

and we start our programme with the <N> command:

N CALL 8000

The whole screen turns white and remains so until you press a key. Then you find yourself in the command input loop of the monitor. You can also calI the programme from Basic (as an in effect games etc.):

G 0066 418E <RETURN>

CALL 8000

The whole screen turns white again. Press any key and you're back to Basic READY. If you now input the NAME, you get into the monitor again, because a breakpoint was set at the adress of the NAME command (418E).

Finally you can store your programme on tape. Prepare your recorder for recording, put a tape in and press PLAY & RECORD. Then input the following:

W8000 8010 8000

Name: TEST <RETURN>

Your programme is written to tape and you can read it back in anytime with the SYSTEM command. Let's try it! First we delete the programme in memory.

C8000 8010 <RETURN.>

Then return to BASIC

Now we rewind the cassette to the beginning of the programme and press the PLAY key. Then input:

```
SYSTEM <RETURN>
*? T <RETURN>

Now your Programme is loaded and shortly appear the characters:
*? <BREAK>
```

```
READY

>CALL 8000
```

Your screen turns white again. You may wonder why the programme could not be started, after loading with the Slash /, although we had given an entry address on saving. The reason is as follows:

We concluded our programme with a RET. The BASIC command CALL does do a subroutine call so that this RET will work perfectly. But the SYSTEM command starts all programmes with a JP i.e. the stack does not contain any return address. So the RET causes the computer to jump anywhere usually to MEM SIZE. So, to start the programme directly after loading, it would have to finish with a JP OO66 for example. This would cause a jump back to BASIC READY. Disadvantage: the little programme can then no longer be called up from a Basic programme, as it always jumps back to BASIC READY and thus interrupts a programme (as does the LIST command.)

V REFERENCE LIST OF ALL COMMANDS

1. KEYBOARD

<CLEAR> New input

< > Cursor one space back

<RETURN> End input (at beginning of a hex number)

<BREAK> Intrrupt an input or command

<SHIF> & <*> Hard copy of screen contents to printer.

2 COMMANDS

A a Start the Direct Assembler at address a

<left arrow> One byte back

<right arrow> One byte forward

<up arrow> One command back

<down arrow> One command forward

<CLEAR> Begin input again

<BREAK> End assembly

<RETURN> Accept command

DM Define Message

DW Define word

DB Define byte

B Set breakpoint for single stepper

<up arrow> Previous address

<down arrow> Next address

<0> <F> change value

<right arrow> cursor forward one character

<left arrow> > cursor back one character

<RETURN> Accept now value

<;> or <+> increase value by 1

< > decrease value by 1 <X> delete entry

<BREAK> end input

C delete addresses entered with B

C a fill memory from a to FFFFH with 00H

C a b fill memory from a to b with OOH

C a b c fill memory from a to b with c

D Register Memory Display switch on/off

E Continue search with the Find Disassembler

E a Input machine code command, which is then searched for from address a

<left arrow> Cursor one space back

<right arrow> Cursor one space forward

<RETURN> Accept command

F Continue search for a byte sequence

F a Continue search from address a

F a b c search for b,c from address a

G Start the programme from the PC

G a Start the programme from address a

G a. b c Start the programme from address a set breakpoints at addresses b and c

H a b print a, b in decimal: sum, difference, Product and quotient in hex.

I a b c (d e) Relocate programme.

A programme in memory between a and b is relocated. c is added to every address between d and e (or between a and b if d and e are not specified)

Relocate without stopping

S step at every command.

<Up arrow> one byte, back

<down arrow> one byte forward

C Stop at every command to be changed

Y execute change

N don’t change

J Start the singlestepper from PC. Options selected with <T> are heeded. Interrupt execution with (BREAK).

K Go on singlestepping until a RET is executed

L (a) Load a System tape. If a is inputted,

M a change memory contents from address a:

<right arrow> Cursor half a byte to the right

<left arrow> Cursor half a byte the left

< > <up arrow> Cursor 8 bytes back

< > <down arrow> Cursor 8 bytes forward

<0> <F> Change half byte

<CLEAR> Switch between ASCII and HEX input

<F1> note address

<F2> calculate offset

<BREAK> end input

N Assemble and execute a command. CALL and RST commands are executed. RET, JP and JR commands change the PC register (when the condition is fulfilled) All other commands do not change the PC. The PC must be in the RAM area (i.e. above 4000H) so that this command will work.

O Ascertain where a programme will be loaded. This command corresponds closely to the <L> command, but nothing is loaded into memory.

P Input addresses of subroutines to be handled by the singlestepper. Input as described for the <S> command.

Q Switch the PC disassembler on/off.(With M and A the surroundings of the edited address are displayed.)

Q a Disassemble from address a. (11 lines)

```
<RETURN>     1 page forward
<   > 1 page back.
< ; >  or <+> 1 command forward
< > I command back
< > byte forward
<down arrow> 1 byte forward
<up arrow> 1 byte back
 <spacer> indicate new address

 R reg a Load register reg with a
```

Admissible register names are:

B,B’,C,C’,D,D’,E,E’,H,H’,L,L’,A,A’,F,F’,BC,BC’,DE,DE’,HL,HL’
AF,AF’,SP,PC’,IX,IY

After inputting the register name, the spacer must be pressed.

S Execute a command (from PC). CALLS and RSTs are executed in single steps.

T Select options for the single stepper:

<up arrow> one line back

<down arrow> one line forward

<Y> or <N> set flag to Yes or No

<0> <F> change a value

<right arrow> cursor one space forward

<left arrow> cursor one space back

<RETURN> accept value

<BREAK> finish

U simulate a RET

V Compare memory areas. Continue with <RETURN>.End with <BREAK>

W a b c Save programme in System format on cassette. Programme begins at address a, ends at b, starts at c.

X Execute a command (from PC). CALLS and RSTs are executed as commands.

Y a b c Shift a memory block a b to c

Z a b Print out a disassembled listing from a to b

Hex/ASCII dump 64 bytes back.

+ or ; Hex/ASCII dump 64 bytes forward.

<up arrow> Decrease PC by 1

<down arrow> Increase PC by 1

<spacer> Jump a command

: Set the PC back one command.

<CLEAR> Blank the central part of the screen. Switch off <D> and <Q>

<left arrow> Hex/ASCII dump one byte back.

<right arrow>   Hex/ASCII dump one byte forward.

.   Display the saved screen until <RETURN> is pressed.

/   Display Hex/ASCII dump full screen.

<;> or <+> 200 bytes forward. < > 200 bytes back.

<BREAK>   Finish

<1> <0>   Single step 1 10 commands. Options   See <T> command.

,   Fill screen buffer (if present) with spaces